PRACTICAS CON ARDUINO: Nivel 1

1. INTRODUCCIÓN

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica.

Un microcontrolador es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida.

El hardware de Arduino consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. Consta de 13 "puertos" o conexiones digitales que pueden ser utilizados cono salidas o entradas (aunque es aconsejable no utilizar la 0 y la 1). Cuando son utilizadas como salidas darán un 0 o un 1, y por tanto, 0 voltios o 5 voltios. Con esta tensión podemos controlar dispositivos electrónicos que consuman poca energía, por ejemplo LED, un zumbador, o cualquier otro cuyo consumo sea inferior a 20 mA. Para controlar un motor, una lámpara, incluso un Relé, es necesario utilizar un transistor.

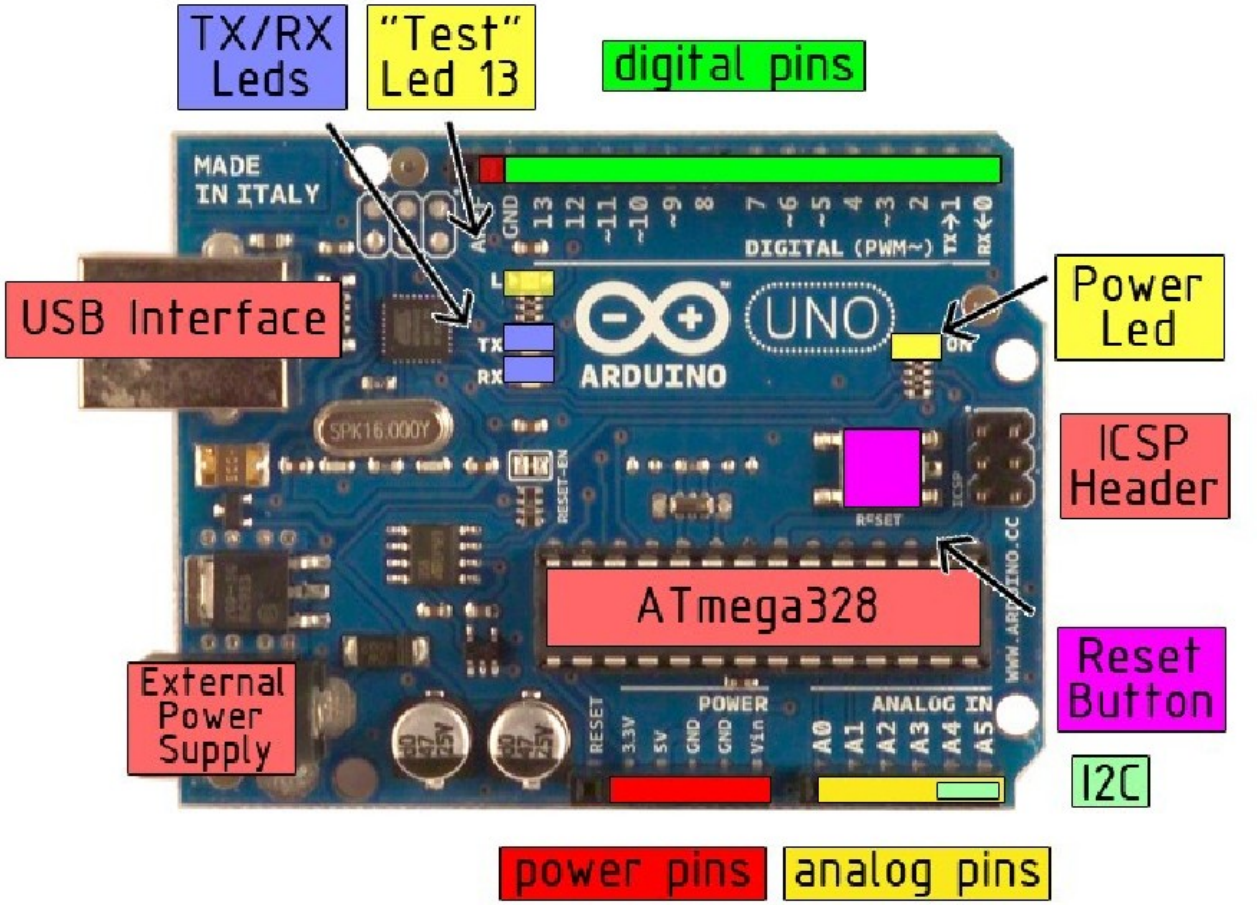
También consta de 5 entradas analógicas que pueden recibir tensiones de 7 a 12 v, procedentes de cualquier sensor que proporcione información del exterior a nuestro Arduino: temperatura, luminosidad, humedad, velocidad, posición,…

La plataforma Arduino se programa mediante el uso de un lenguaje propio basado en el popular lenguaje de programación de alto nivel Processing. Sin embargo, es posible utilizar otros lenguajes de programación y aplicaciones populares en Arduino. Algunos ejemplos son: Java, Flash (mediante ActionScript), Processing, Pure Data, MaxMSP (entorno gráfico de programación para aplicaciones musicales, de audio y multimedia), VVVV (síntesis de vídeo en tiempo real), Adobe Director, Pitón, Ruby, C, C++ (mediante libSerial o en Windows), Cocoa/Objective-C (para Mac OS X), Linux TTY (terminales de Linux), 3DVIA Virtools (aplicaciones interactivas y de tiempo real), SuperCollider (síntesis de audio en tiempo real), Instant Reality (X3D), Visual Basic .NET, VBScript, Gambas, Php,…

Esto es posible debido a que Arduino se comunica mediante la transmisión de datos en formato serie que es algo que la mayoría de los lenguajes anteriormente citados soportan. Para los que no soportan el formato serie de forma nativa, es posible utilizar software intermediario que traduzca los mensajes enviados por ambas partes para permitir una comunicación fluida. Es bastante interesante tener la posibilidad de interactuar Arduino mediante esta gran variedad de sistemas y lenguajes puesto que dependiendo de cuales sean las necesidades del problema que vamos a resolver podremos aprovecharnos de la gran compatibilidad de comunicación que ofrece.

Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software del ordenador (por ejemplo: Macromedia Flash, Processing, Max/MSP, Pure Data). Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente.

Al ser open-hardware, tanto su diseño como su distribución es libre. Es decir, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin haber adquirido ninguna licencia.



2. PROGRAMACIÓN BÁSICA

La estructura básica del lenguaje de programación de Arduino es bastante simple y se compone de al menos dos partes. Ambas funciones son necesarias para que el programa trabaje.

Hay que ser muy cuidadoso y escribir los "comandos" exactamente, reSpetando mayúsculas y minúsculas y colocando " **;** " al finalizar la línea de comando.

La estructura básica es esta:

`void setup()`
`{`
`  Configuración;`
`}`
	En el setup, entre los corchetes { } tenemos que indicar los datos de configuración del programa. Por ejemplo que "pin" o puerto vamos a utilizar, cuales son entradas y cuales salidas, cuales son las variables…
	El setup se ejecuta sólo una vez al comenzar el programa, mientras que lo que pongamos en el loop se va a repetir indefinidamente.

`void loop()`
`{`
`  Programa que se repite;`
`}`
	La función bucle (loop) siguiente contiene el código que se ejecutara continuamente (lectura de entradas, activación de salidas, etc) Esta función es el núcleo de todos los programas de Arduino y la que realiza la mayor parte del trabajo.

PRÁCTICA 1. Activación de una salida digital.

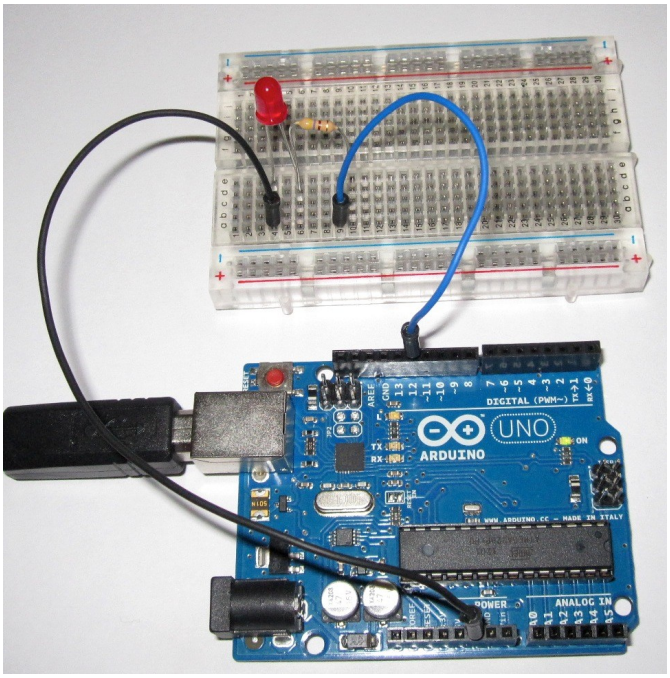
El LED conectado a la salida digital nº 12 parpadea.

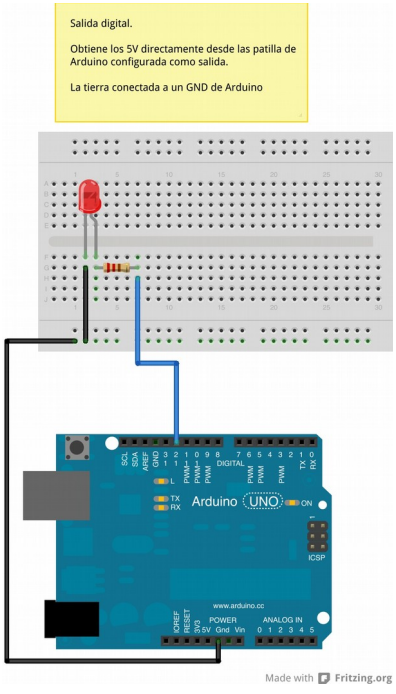
pinMode(patilla, modo). El modo puede ser INPUT (que es entrada) o OUTPUT (que es salida).

digitalWrite(patilla,valor). Escribe en una patilla un valor HIGH (que es alto) o LOW.

void setup()
{
pinMode(12, OUTPUT);
}
void loop()
{
digitalWrite(12, HIGH);
delay(500);
digitalWrite(12, LOW);
delay(100);
}
	Dentro del **setup** (entre los corchetes), colocamos la orden pinMode. Con esta orden indicamos que la salida 12 será una Salida.
	Dentro del **loop** (entre los corchetes)
	digitalWrite
	"escribe en la salida digital 12 un valor 1"
	Ojo!! Después de cada orden hay que escribir " **;** "
	delay
	Hace esperar al programa 500 milisegundos (o sea 0,5 segundos)
	Repite la orden y escribe un valor 0 en el pin 12. Espera 0,1 segundo.
	Repite el programa indefinidamente.

Una vez escrito el programa pulsamos el botón verificar, para detectar errores de escritura. Si no hay ninguno conectamos arduino al ordenador y pulsamos cargar para que lo ejecute.A los programas se les suele poner un texto que explica como funcionan. El próximo programa que realicemos lo tendrá. Pero hay que tener mucho cuidado y escribir el texto adecuadamente para que no de error.





Podemos hacer una mejora de este programa para que quede más "profesional", utilizando **variables**. Esto será lo más habitual más adelante. Las variables son un valor que arduino guarda en su memoria y que después podemos utilizar.

int Led=12;
int t=500;
void setup()
{
pinMode(Led, OUTPUT);
}
void loop()
{
digitalWrite(Led, HIGH);
delay(t);
digitalWrite(Led, LOW);
delay(t);
}
	Ahora le decimos a Arduino que la variable "Led" es 12 y la variable "t" es 500.
	Dentro del **setup**, colocamos la orden pinMode hace que "Led" sea una salida. Como hemos dicho que "Led =12" querrá decir que el pin 12 del Arduino será una salida de información.
	Dentro del **loop**
	digitalWrite
	Escribe en "Led" un valor 1, o sea pon a 5 v la salida 12
	delay
	Hace esperar al programa t milisegundos (como le hemos dicho que t=500, se espera 500 milisegundos = 0,5 segundos)
	Repite la orden y escribe un valor 0 en el Led. Espera t milisegundos.
	Repite el programa indefinidamente.

PRÁCTICA 2. Activación de 3 salidas digitales de forma consecutiva.Creación de variables.

Se encienden alternativamente 3 LED.

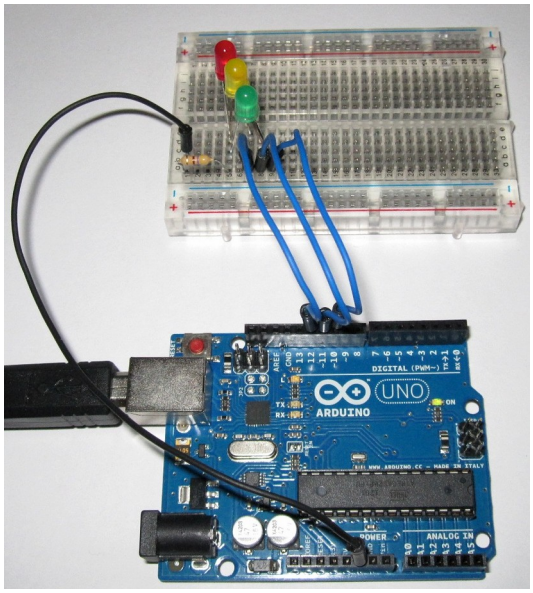
Vamos a crear tres **variables** que asignaremos a cada una de las patillas donde hemos conectado los LED.

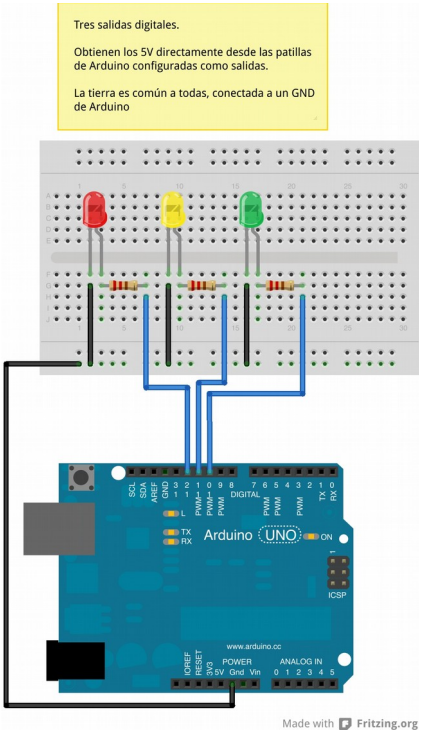
Programa con las explicaciones iniciales que deben ir /* explicación */

Si queremos hacer alguna explicación en la linea de comando debe de ser después de //

```
/*
02 Salidas Digitales
Los tres led parpadean formando una secuencia Rojo, Amarillo y Verde.
Instrucciones :
Debemos utilizar el punto y coma " ; " al final de cada declaración separando los elementos del probrama.
Con la orden int creamos una variable, por ejemplo la variable "rojo" y le asignamos un valor, por ejemplo valor "12".
Si la variable queremos que esté disponible para todo el programa debe de crearse ANTES del setup(). También podemos
crear variables que sólo afecten a una parte del programa.
Es necesario escribir las ordenes correctamente. NO pinmode sino pinMode.
Si está bien escrito aparece en color marrón.
Conexiones:
pin 12 : LED Rojo
pin 11 : LED Amarillo
pin 10 : LED Verde
Crearemos las variables rojo, amarillo y verde y les asignaremos el número de pin.

José Antonio González
*/


int rojo = 12;        // Crea una variable llamada "rojo" y le asigna un valor 13
int amarillo = 11;
int verde = 10;

void setup()
{
  pinMode(rojo, OUTPUT);     // Indica que la variable " rojo " es una salida de arduino, y por tanto el pin 13 es una salida.
  pinMode(amarillo, OUTPUT);
  pinMode(verde, OUTPUT);
```

```
}

void loop()
{
  digitalWrite(rojo, HIGH);        // Escribe en la variable "rojo" un valor alto, un "1".
  delay(500);                      // Espera 0,5 segundos
  digitalWrite(rojo, LOW);         // Escribe en la variable "rojo" un valor bajo, un "0".


  digitalWrite(amarillo, HIGH);
  delay(500);
  digitalWrite(amarillo, LOW);


  digitalWrite(verde, HIGH);
  delay(500);
  digitalWrite(verde, LOW);
}
```



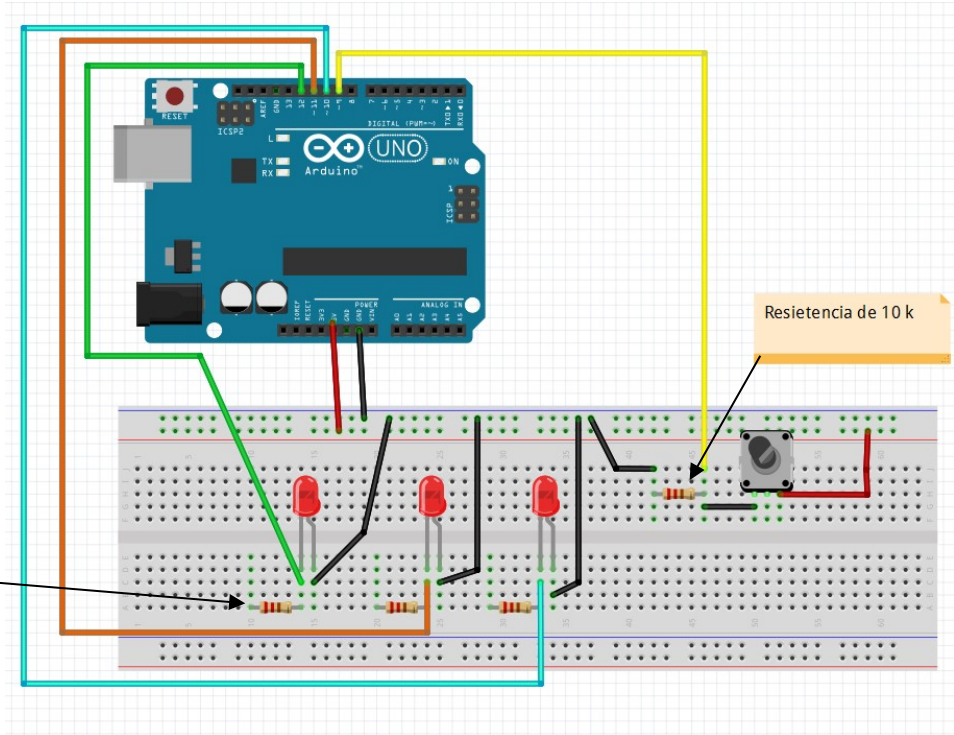


ORDEN IF

La orden IF es una estructura **condicional:**
si se cumple la **condición** se ejecuta { programa}. Si no se cumple la condición se para.
if (condición**) {**...**}**

PRÁCTICA 3. Pulsador. Activación de 3 salidas digitales de forma consecutiva a partir de que presionemos un pulsador.

```
int rojo = 12;          // Crea una variable llamada "rojo" y le asigna un valor 13
int amarillo = 11;
int verde = 10;
int tiempo = 500;
int pulsador = 9;
void setup()
{
  pinMode(rojo, OUTPUT);     // Indica que la variable " rojo " es una salida de arduino, y por tanto el pin 13 es una salida.
  pinMode(amarillo, OUTPUT);
  pinMode(verde, OUTPUT);
  pinMode(pulsador, INPUT);
}
void loop()
{
if (digitalRead(pulsador) == HIGH)   // lee el valor que tiene el pin 9, si el valor es IGUAL a 1 entonces ejecuta el
programa, de lo contrario no hace nada y espera. Para que se repita tendremos que volver a pulsar.
{

  digitalWrite(rojo, HIGH);        // Escribe en la variable "rojo" un valor alto, un "1".
  delay(tiempo);                   // Espera el tiempo establecido en la variable tiempo
  digitalWrite(rojo, LOW);       // Escribe en la variable "rojo" un valor bajo, un "0".


  digitalWrite(amarillo, HIGH);
  delay(tiempo );
  digitalWrite(amarillo, LOW);


  digitalWrite(verde, HIGH);
  delay(tiempo);
  digitalWrite(verde, LOW);
}
}
```



EXPERIMENTA TU SÓLO Y RESUELVE ESTOS PROBLEMAS

PRÁCTICA 4. Regulación de un cruce con semáforos.

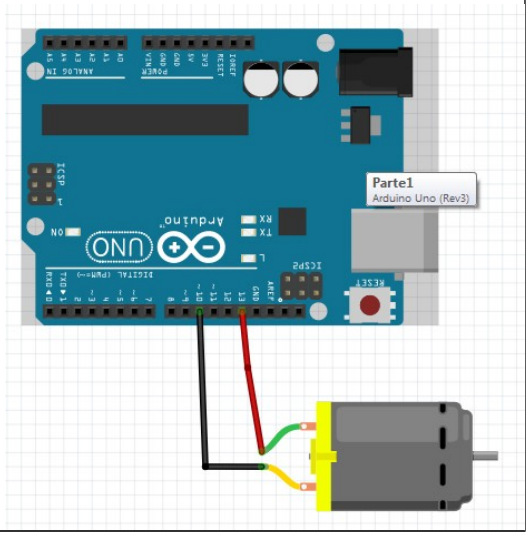
Tenemos dos semáforos con una luz roja, otra amarilla y otra verde. Llamaremos rojo1, amarillo1 y verde1 a las luces de uno de los semáforos y rojo2, amarillo2 y verde2 a las luces del otro. Piensa la secuencia lógica de funcionamiento del sistema escríbela y posteriormente realiza el programa. Utiliza los siguientes pines, creando las variables correspondientes.

pin 13 : LED Rojo1	pin 7 : LED Rojo2
pin 12 : LED Amarillo1	pin 6 : LED Amarillo2
pin 11 : LED Verde1	pin 5 : LED Verde2

PRÁCTICA 5. Motor que gira a derechas y a izquierdas

Los motores de corriente continua que utilizamos en clase giran en una u otra dirección según conectemos el positivo (+5 v) en uno u otro cable. Con este programa haremos que el motor gire en una dirección y luego en la otra.

```
int rojo = 12;          // Crea una variable llamada "rojo" y le asigna un valor 12
int negro = 10;
void setup()
{
  pinMode(rojo, OUTPUT); // Indica que la variable"rojo" es una salida de arduino, y por tanto el pin 12 es una salida.
  pinMode(negro, OUTPUT);
}
void loop()
{
  digitalWrite(rojo, HIGH);        // EL MOTOR GIRA A DERECHAS. Escribe en la variable "rojo" un valor alto, un "1".
  digitalWrite(negro, LOW);     // Escribe en la variable "negro" un valor alto, un "0".
  delay(2000);                  // Espera 2 segundos
  digitalWrite(rojo, LOW);      // Escribe en la variable "rojo" un valor bajo, un "0".
 delay(500);                    // Espera 0,5 segundos

  digitalWrite(negro, HIGH); // EL MOTOR GIRA A IZQUIERDAS.
digitalWrite(rojo, LOW);
  delay(2000);
  digitalWrite(negro, LOW);
  delay(500);
 }
```



PRÁCTICA 6. Crear nuestras FUNCIONES. Motor gira a derechas y a izquierdas.

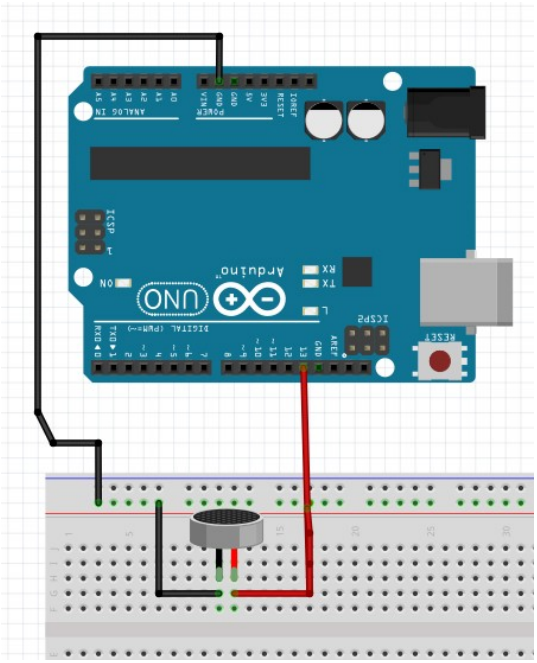
Arduino tiene sus funciones como pinMode, digitalWrite,…pero nosotros también podemos crear nuestras propias FUNCIONES. Cuando las escribimos el programa realizará el conjunto de instrucciones que nosotros le indiquemos. Crearemos dos funciones una Motor_derecha y otra Motor_izquierda. Lo que harán es que el motor gire en una u otra dirección un tiempo. El programa es igual que el anterior pero con esta modificación. Esto podría servir para controlar los motores de un coche, por ejemplo.

```
int rojo = 12;          // Crea una variable llamada "rojo" y le asigna un valor 12
int negro = 10;
void setup()
{
   pinMode(rojo, OUTPUT); // Indica que la variable "rojo" es una salida de arduino, y por tanto el pin 12 es una salida.
  pinMode(negro, OUTPUT);
}
void loop()
{
  Motor_derecha();          // ejecuta la función MOTOR GIRA A LA DERECHA
  delay(2000);
     Motor_derecha();          // ejecuta la función MOTOR GIRA A LA DERECHA de nuevo para ver la diferencia.
  Motor_izquierda();          // ejecuta la función MOTOR GIRA A LA IZQUIERDA
 }
void Motor_derecha()       // CREAMOS LA FUNCIÓN Motor_derecha
{
 digitalWrite(rojo, HIGH);
 digitalWrite(negro, LOW);
  delay(2000);
  digitalWrite(rojo, LOW);
 delay(500);
}

void Motor_izquierda()      // CREAMOS LA FUNCIÓN Motor_izquierda
{
 digitalWrite(negro, HIGH);
  digitalWrite(rojo, LOW);
  delay(2000);
  digitalWrite(negro, LOW);
 delay(500);
}
```

PRÁCTICA 7. SOS con un zumbador

Se trata de un zumbador que en código morse (pitidos largos y cortos) emite una palabra, en nuestro caso SOS. En código morse la S se reproduce con 3 señales de corta duración (3 pitidos cortos de 100 ms para la S) y la O con tres señales de larga duración tres pitidos largos de 400 ms para la O). Entre letra y letra dejamos 500 ms, y entre SOS y SOS tiene que pasar un tiempo mayor de 2000 ms (2 sg).

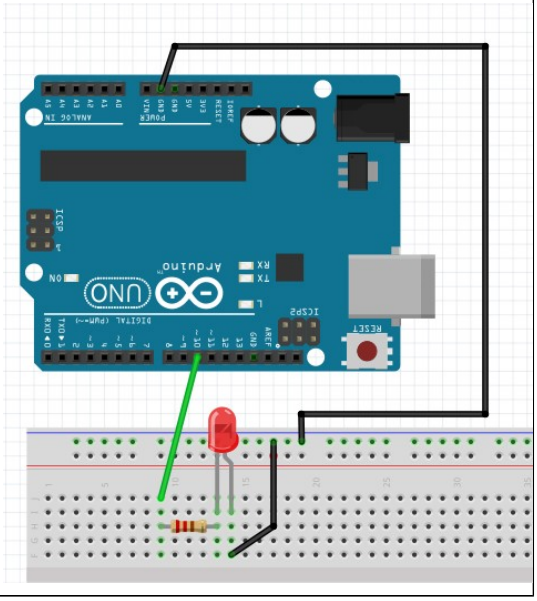
```
int corto=100;    //Declara la variable de argumento entero "corto" y la inicializa con el valor 100  (letra S)
int espacio=100    // Espacio ente pitito y pitido
int pausa=500;    //tiempo entre letra y letra
int largo=400;   //variable de argumento entero "largo" y la inicializa con el valor 300 (letra O)
int espera=2000;  //variable argumento entero "espera" y la inicializa con el valor 1000 (tiempo entre SOS - SOS)
int n=0;
int zumbador=13;    //PIN digital al que conectamos el zumbador
void setup()      //comienza la configuración
{
pinMode(zumbador,OUTPUT);
}
void loop()
{
digitalWrite(zumbador, HIGH);   // comienza la S
delay(corto);
digitalWrite(zumbador,LOW);
delay(espacio);
digitalWrite(zumbador, HIGH);
delay(corto);
digitalWrite(zumbador,LOW);
delay(espacio);
digitalWrite(zumbador, HIGH);
delay(corto);
digitalWrite(zumbador,LOW);
delay(pausa);
digitalWrite(zumbador, HIGH);   // comienza la O
delay(largo);
digitalWrite(zumbador,LOW);
delay(espacio);
digitalWrite(zumbador, HIGH);
delay(largo);
digitalWrite(zumbador,LOW);
delay(espacio);
digitalWrite(zumbador, HIGH);
delay(largo);
digitalWrite(zumbador,LOW);
delay(pausa);
digitalWrite(zumbador, HIGH);   // comienza la S
delay(corto);
digitalWrite(zumbador,LOW);
delay(espacio);
digitalWrite(zumbador, HIGH);
delay(corto);
digitalWrite(zumbador,LOW);
delay(espacio);
digitalWrite(zumbador, HIGH);
delay(corto);
digitalWrite(zumbador,LOW);
delay(espera);
}
```



PRÁCTICA 8. Encendido de un Led un número de veces

Haremos que se encienda y se apague un led 10 veces y luego se pare y no haga nada

```
int i=0;
int led=10;
void setup()
{
 pinMode(led,OUTPUT);
}
void loop()
{
if(i<10)     // si la variable i tiene un valor menor que 10 se ejecuta el programa, si es mayor para
{
digitalWrite(led,HIGH);     //enciendo el led poniendo a 1 el pin 10
delay(1000);                  //lo deja encendido 1 sg
digitalWrite(led,LOW);       //apaga el led
delay(1000);                 //lo deja apagado 1 seg
i=i+1;                          // le suma 1 al valor inicial de i
}
}
```



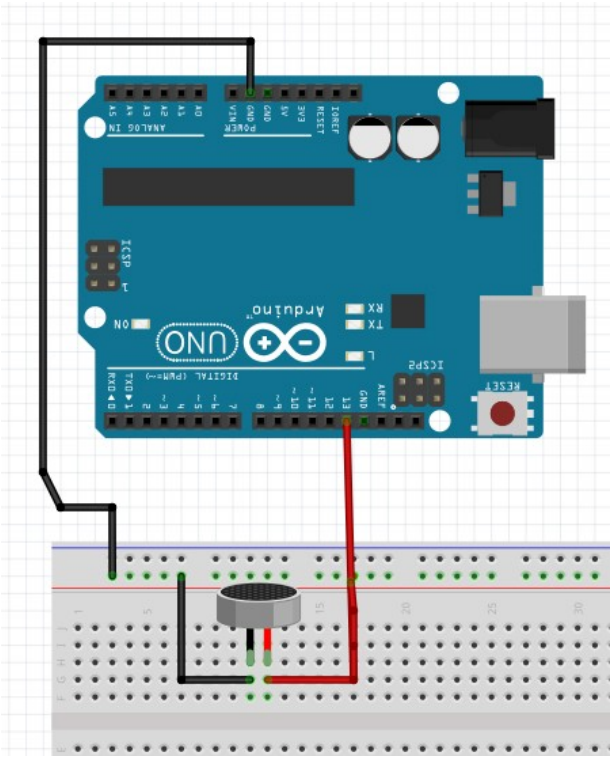
REPASA LO QUE HAS APRENDIDO:

Sustituye el led por un zumbador y añade un pulsador como viste en la práctica 3. Cambia el circuito de modo que cada vez que presiones un pulsador, suene 5 veces un zumbador.

PRÁCTICA 9. SOS con un zumbador que repite 5 veces utilizando FUNCIONES

Vamos a utilizar la orden IF para que un zumbador emita 6 veces SOS. Crearemos 2 funciones para producir la letra S y la letra O.

```
int corto=100;    //Declara la variable de argumento entero "corto" y la inicializa con el valor 100  (letra S)
int espacio=100    // Espacio ente pitito y pitido
int pausa=500;    //tiempo entre letra y letra
int largo=400;   //variable de argumento entero "largo" y la inicializa con el valor 300 (letra O)
int espera=2000;  //variable argumento entero "espera" y la inicializa con el valor 1000 (tiempo entre SOS - SOS)
int n=0;
int zumbador =13;    //PIN digital al que conectamos el zumbador
void setup()      //comienza la configuración
{
pinMode(zumbador,OUTPUT);
}
void loop()
{
if (n<6)          // si n es menor que 6 ejecuta el programa, si no se para porque no hemos indicado ELSE
{
S();
delay(pausa);
O();
delay(pausa);
S();
delay(espera);
n=n+1;                  // aumenta el valor de la variable n en una unidad
}
}
void S()
{
digitalWrite(zumbador, HIGH);   // comienza la S
delay(corto);
digitalWrite(zumbador,LOW);
delay(espacio);
digitalWrite(zumbador, HIGH);
delay(corto);
digitalWrite(zumbador,LOW);
delay(espacio);
digitalWrite(zumbador, HIGH);
delay(corto);
digitalWrite(zumbador,LOW);
  }
void O()
{
digitalWrite(zumbador, HIGH);   // comienza la O
delay(largo);
digitalWrite(zumbador,LOW);
delay(espacio);
digitalWrite(zumbador, HIGH);
delay(largo);
digitalWrite(zumbador,LOW);
delay(espacio);
digitalWrite(zumbador, HIGH);
delay(largo);
digitalWrite(zumbador,LOW);
}
```



ORDEN IF

La orden IF es una estructura **condicional:**
si se cumple la **condición** se ejecuta { programa}. Si no se cumple la condición se para.

if (condición**) {**...**}**

A la hora de comprobar si una condición se cumple o no, podemos utilizar los siguientes **operadores de comparación** dentro del paréntesis:

`x == 3`	si `x` es igual a 3
`x != 3`	si `x` es distinto de 3
`x < 3`	si `x` es menor que 3
`x > 3`	si `x` es mayor que 3
`x <=3`	si `x` es menor o igual que 3
`x >= 3`	si `x` es mayor o igual que 3
`x < 2 || x > 5`	si `x` es menor que 2 **o** mayor que 5
`x == 4 && z <= 8`	si `x` es igual a 4 **y** `z` es menor o igual que 8
`! x > 0`	si `x` **no** es mayor que 0 (si es menor o igual que 0)

¡Importante! No confundir = (que es para asignar) con == (que es para comparar).
Para modificar el valor de una variable podemos utilizar las siguientes operaciones:

```
i=i+5     //el valor de i se incrementa en 5
i=i+1     //el valor de i se incrementa en 1
i++       //el valor de i se incrementa en 1 (sólo para incremento +1)
i=i-1     //el valor de i disminuye en 1
i--       //el valor de i disminuye en 1 (sólo para incremento -1)
i=i*3     //el valor de i se multiplica por 3
i=i/2     //el valor de i se divide entre 2
```

Una variante a esta estructura es la formada por: si se cumple la condición hace lo que indica el programa 1 y si no se cumple lo que indique el programa 2 (else).

if (condición**) {***programa1*…**} else {** *programa2***}**

Permite que el programa coja uno de los dos caminos: si se cumple la condición (que será lo que acompañe al `if`), o si no lo cumple (que será lo que acompañe a `else`).

Haremos que se encienda y se apague un led 10 veces y luego se ejecute el **else** y el led quede encendido indefinidamente

```
int i=0;
int led=10;
void setup()
{
 pinMode(led,OUTPUT);
}
void loop()
{
if(i<10)    // si la variable i tiene un valor menor que 10 se ejecuta el programa, si es mayor para
{
digitalWrite(led,HIGH);    //enciendo el led poniendo a 1 el pin 10
delay(1000);                //lo deja encendido 1 sg
digitalWrite(led,LOW);      //apaga el led
delay(1000);               //lo deja apagado 1 seg
i=i+1;                      // le suma 1 al valor inicial de i
}
else
{
digitalWrite(led,HIGH);
}
```

PRÁCTICA 10. Luz que avanza por 5 Led utilizando la orden IF

Con este programa van a encenderse 5 Led de forma consecutiva. El proceso se repite indefinidamente.

Vamos a utilizar 5 led rojos conectados a las salidas 7,8,9,10 y11.

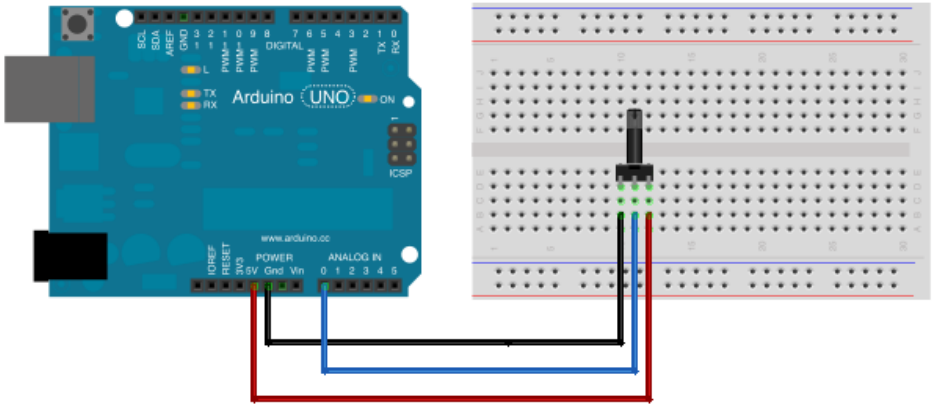
int s=7;
int t=1000;
void setup()
{ pinMode(7, OUTPUT);
pinMode(8, OUTPUT);
pinMode(9, OUTPUT);
pinMode(10, OUTPUT);
pinMode(11, OUTPUT); }
void loop()
{
IF (s=7; s<=11; s+1)
{ digitalWrite(s,HIGH);
digitalWrite(s-1,LOW);
delay(t);
}
digitalWrite(11,LOW);
}
	Creamos una variable llamada “s” que en un principio tendrá un valor 7. Este valor irá variando con la orden FOR.
	setup
	Indicamos que los pin 7, 8, 9, 10 y 11 son salidas.
	colocamos la orden pinMode hace que “Led” sea una salida. Como hemos dicho que “Led =12” querrá decir que el pin 12 del Arduino será una salida de información.
	
	loop
	for (inicio; condición ;incremento) {….}
	Hará todo lo que hay en el bucle for hasta que no se cumplan las condiciones.
	Comienza encendiendo la salida 7 y apagando la 6 (que no existe). Espera un tiempo t. Le suma 1.
	Ahora enciende la salida 8 y apaga la 7. Como 8 es <=11, continúa.
	Enciende la 9 y apaga la 8. Enciende la 10 y apaga la 9. Enciende la 11 y apaga la 10.
	Cuando llega a 12 que no es <=11. Sale del bucle FOR apaga la salida 11.
	Todo vuelve a comenzar y llama s=7

PRÁCTICA 11. Medición de la tensión de un potenciómetro. Visualización en el ordenador.

En este proyecto vamos a visualizar los valores de tensión en un potenciómetro y para conocerlos, vamos a desplegarlos en nuestro ordenador con la orden Monitor Serial del IDE.

El potenciómetro es una resistencia variable que nos proporcionará los valores analógicos de un voltaje, valores entre el máximo y el mínimo del voltaje al que está conectado. Estos valores los podremos obtener utilizando la función analogRead() en uno de los puertos analógicos del arduino.

Para utilizarlo, conectamos **la pata de un extremo a Voltios, la del otro extremo a 0 v (GND)** y **la pata central, donde tenemos la resistencia variable, la conectamos al pin analógico 0 del arduino.** Es en esta pata central donde mediremos los valores del voltaje que nos da el potenciómetro.



Vamos a usar la función **analogRead()** para leer los valores analógicos que nos proporciona el potenciómetro. A analogRead() hay que indicarle que pin hay que leer o medir. El valor que mide lo convierte en uno entre 0 y 1023. La conversión la hace el microcontrolador utilizando un número de 10 bits, donde el mínimos será entonces un 0 para 0 volts, el máximo 1023, equivalente a 5 Voltios y por ejemplo, 512 equivalente a 2.5 Voltios.

Para transmitir los datos a la computadora usaremos la **biblioteca Serial**. La inicializamos en la función setup con la velocidad a la que queremos transmitir la información, Serial.begin(9600);

Para enviar datos del arduino hacia la computadora usamos la función Serial.println(). Cuando giremos la varilla del potenciómetro, cambia su resistencia y este cambio se verá en el ordenador.

```
int pinSensor = A0;          // pin del sensor analogico, con un potenciometro
int valorAnalogico  = 0;   // variable para guardar el valor leido del sensor

void setup()
{
pinMode(pinSensor, INPUT); // Aunque no es necesario en este caso incializa el pin del boton como de entrada
Serial.begin(9600);              // Inicializa la comunicacion serial
}

void loop(){
valorAnalogico = analogRead(pinSensor);          //  lee el valor del sensor
Serial.print("Valor del sensor analogico = ");       //  Escribe en el monitor serial este texto
Serial.println(valorAnalogico);                          //  Escribe el valor almacenado en la variable “valorAnalógico” en el monitor
delay(1000);                                                 // espera 1000 milisegundos para leer y enviar la siguiente lectura del sensor
  }
```

PRÁCTICA 12. Control de Led con un potenciómetro.

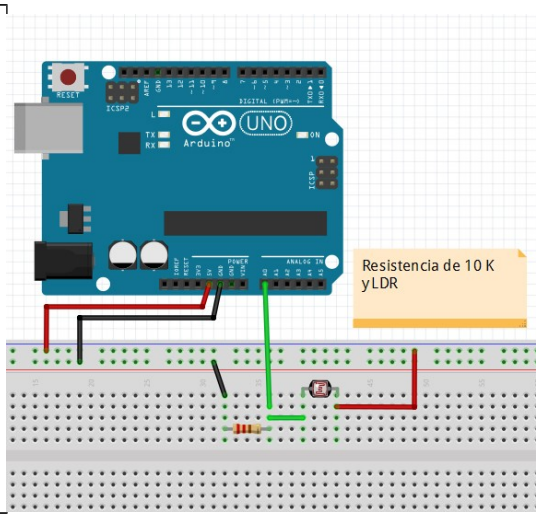
Utilizamos 4 Led conectados a los pin 8,9,10 y 11. Si el valor de la tensión del potenciómetro es 0 todos los led estarán apagados, si el valor está entre 0 y 400, se enciende el led conectado al pin 8, si está entre 400 y 800 el led 9, si está entre 800 y 1023 el led 10 y si el valor de la tensión es igual a 1023 se encienden todos los led a la vez.

```
int i = 0;
int pinPotenciometro =A0;
int valorPotenciometro = 0;

void setup()
{
pinMode(pinPotenciometro, INPUT);     // pinPotenciometro es una entrada
pinMode(8, OUTPUT);                   // pin 8 es una salida
pinMode(9, OUTPUT);                   // pin 9 es una salida
pinMode(10, OUTPUT);                  // pin 10 es una salida
pinMode(11, OUTPUT);                  // pin 11 es una salida

Serial.begin(9600);                   // inicializa monitor serial
}
void loop()
{
valorPotenciometro = analogRead(pinPotenciometro);    //asigna a la variable "valorPotenciometro el valor que lee en el
                                                       pin A0 llamado pinPotenciómetro

Serial.println(valorPotenciometro);     // escribe en el monitor serial el valor del potenciómetro
delay(1000);
if (valorPotenciometro == 0) apagarLeds();   // si el valor del potenciómetro es 0 ejecuta la orden "apagarLeds"
if (valorPotenciometro > 0 && valorPotenciometro < 400) digitalWrite(8, HIGH); // si el valor esta entre 0 y 400
                                                                                  enciende pin 8
if (valorPotenciometro >= 400 && valorPotenciometro < 800) digitalWrite(9, HIGH);

if (valorPotenciometro >= 800 && valorPotenciometro < 1023) digitalWrite(10, HIGH);

if (valorPotenciometro == 1023) encenderLeds(); // si el valor del potenciómetro es 1023 ejecuta la orden "encenderLeds"
}
void apagarLeds()
{
digitalWrite(8, LOW);
digitalWrite(9, LOW);
digitalWrite(10, LOW);
digitalWrite(11, LOW);

}
void encenderLeds()
{
digitalWrite(8, HIGH);
digitalWrite(9, HIGH);
digitalWrite(10, HIGH);
digitalWrite(11, HIGH);

}
```

PRÁCTICA 13. LDR para medir la luz. Visualizar en el ordenador el valor de esta luz.

Una LDR es una resistencia electrónica que varía su valor en función de la luz que recibe. Vamos a utilizarla para medir la luz ambiente. Para poder ver cual es su valor utilizaremos remos la orden "*Serial.begin(9600)*;" que permite comunicarse nuestro arduino con el PC a una velocidad de 9600 bits/segundo y ver la medida de la luz. Realizaremos un montaje parecido al de la práctica 3, conectando la LDR en serie con una resistencia de 10 K (marrón, negro, naranja), como aparece en la imagen. La señal que indica la medida de la luz entrará en arduino por el pin 0 que es uno de las entradas analógicas. Para que varíe podemos cubrirla con el dedo o enfocarla a una lámpara.

```
int ldr=0;   //conectamos la ldr a la entrada analógica 0
int luz=0;

void setup()
{
  pinMode(ldr, INPUT);
  Serial.begin(9600);
}
void loop()
{
  luz=analogRead(ldr);
  Serial.print("el valor de luz es...");
  Serial.println(luz);
  delay(1000);
}
```



PRÁCTICA 14. LDR con visualización y activación de señal de alarma con un zumbador.

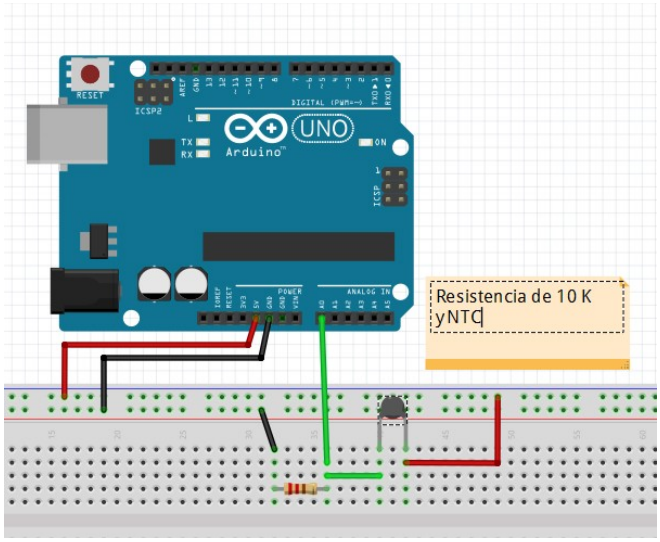
Realiza un montaje similar al de la práctica anterior pero añade un led conectado al pin 10. Cuando la luz sea menor de 500, debe encenderse un led. Si la luz es superior a 500 el led se apaga. Después puedes sustituir el led por un zumbador que haga lo mismo y nos avise cuando la luz es escasa.

```
int ldr=0;   //conectamos la ldr a la entrada analógica 0
int luz=0;   //asignamos a la variable "luz" un valor inicial 0
int led=10; //conectamos la el led a la entrada digital10
void setup()
{
  pinMode(ldr, INPUT);  // definimos las entradas y salidas y activamos el monitor serial del ordenador
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}
void loop()
{
  luz=analogRead(ldr);
  Serial.print("el valor de luz es...");
  Serial.println(luz);
   delay(1000);
   if(luz>=0 && luz < 500)   // si el valor de la luminosidad está comprendido entre 0 y 500 enciende led
  {
  digitalWrite(led,HIGH);
  }
    else                    // Si no está comprendido entre 0 y 500, entonces apaga el led
  {
  digitalWrite(led,LOW);
  }  }
```

PRÁCTICA 15. Medida de la temperatura con una resistencia NTC con visualización serial

Las NTC son resistencias que varían su valor con la temperatura. Conforme aumenta la temperatura disminuye su resistencia. Tendremos un valor numérico que es proporcional a la temperatura que medimos. Una lectura de 500 equivale a 20 º C. Por lo tanto si multiplicamos el número que mide por 0 ´04 tendremos la temperatura en º C.

```
int ntc=0;      //conectamos la resistencia NTC a la entrada analógica 0
int temperatura=0;
int grados=0;
void setup()
{
  pinMode(ntc, INPUT);
  Serial.begin(9600);
}
void loop()
{
  temperatura=analogRead(ntc);
    grados=  0.04*temperatura;     // multiplicamos por 0.04 para
          tener la temperatura en º C
  Serial.print("el valor de la temperatura es...");
  Serial.println(grados);
   delay(1000);
   }
```



PRÁCTICA 16. Medida de la temperatura con una resistencia NTC con visualización serial, activación de señal de alarma y puesta en marcha de un motor.

Realiza un montaje similar al anterior pero añade un zumbador conectado al pin 10 y un motor al pin 11. Cuando la temperatura supere los 25º C debe sonar el zumbador y ponerse en marcha un motor para enfriar.

```
int ntc=0;              //conectamos la resistencia NTC a la entrada analógica 0
int temperatura=0;
int grados=0;
int zumbador=10;
int motor=11;
void setup()
{
  pinMode(ntc, INPUT);
  pinMode(zumbador, OUTPUT);
  pinMode(motor,OUTPUT);
  Serial.begin(9600);
}
void loop()
{
  temperatura=analogRead(ntc);
  grados= 0.04*temperatura;                    // multiplicamos por 0.04 para tener la temperatura en º C
  Serial.print("el valor de la temperatura es...");
  Serial.println(grados);
   delay(1000);
   if(grados > 25)   //Si la temperatura aumenta por encima de 25 debe sonar la alarma y ponerse en marcha el motor
  {
  digitalWrite(zumbador,HIGH);
  digitalWrite(motor, HIGH);
  }
    else
  {
  digitalWrite(zumbador,LOW);
 digitalWrite(motor, LOW);
  }
   }
```

PRÁCTICA 17. Medir distancias con ultrasonidos utilizando el puerto serial del PC

Vamos a utilizar el emisor de ultrasonidos para medir la distancia a objetos y haremos que las mediciones aparezcan en nuestro ordenador. Para ello utilizaremos la orden "*Serial.begin(9600)*;" que permite comunicarse nuestro arduino con el PC a una velocidad de 9600 bits/segundo.

En primer lugar será necesario cargar la **librería** que controla los sensores y los motores.

¿Que es una librería de arduino?

Las librerías son colecciones de código que facilitan la interconexión de sensores, pantallas, módulos electrónicos, etc. El entorno de arduino ya incluye algunas librerías de manera que se facilite, por ejemplo, mostrar texto en pantallas LCD. Existen cientos de librerías desarrolladas por terceros en Internet, que nos ayudarán a conectar prácticamente cualquier dispositivo a nuestras tarjetas con arduino. Las librerías normalmente incluyen los siguientes archivos: Un archivo .cpp (código de C++), un archivo .h o encabezado de C,…

Para cargar una librería que hemos encontrado en Internet en arduino: sketch ␣ importar librería ␣ Add library. Si ya la hemos utilizado antes la tendremos en la lista y sólo tenemos que seleccionarla para que esté disponible para nuestro programa.

#include <Ultrasonic.h> Ultrasonic ultrasonic(9,8,30000);	// conectamos el Trig al pin 9 y el Echo al pin 8, el número 30000 son los ms máximos que mide y equivale a unos 5 metros. Cuanto más cerca esté el objeto menor será el tiempo de regreso del ultrasonido.
void setup() { Serial.begin(9600); }	//comenzamos la comunicación serial y marcamos como velocidad de comunicación 9600 bits/seg

void loop()
{
Serial.print("distancia = ");
Serial.print(ultrasonic.Ranging(CM));
Serial.println(" cm");
delay(1000);
}
	// imprime un texto entrecomillado (distancia=)
	//imprime el valor de la variable distancia al objeto
	// imprime un texto entrecomillado (cm) print LN significa que cambie de linea en cada lectura y no lo escriba todo en una línea. Eso nos facilita la visualización de los datos.
	// Espera 1 segundo.

PRÁCTICA 18. Uso de una pantalla LCD de cristal líquido

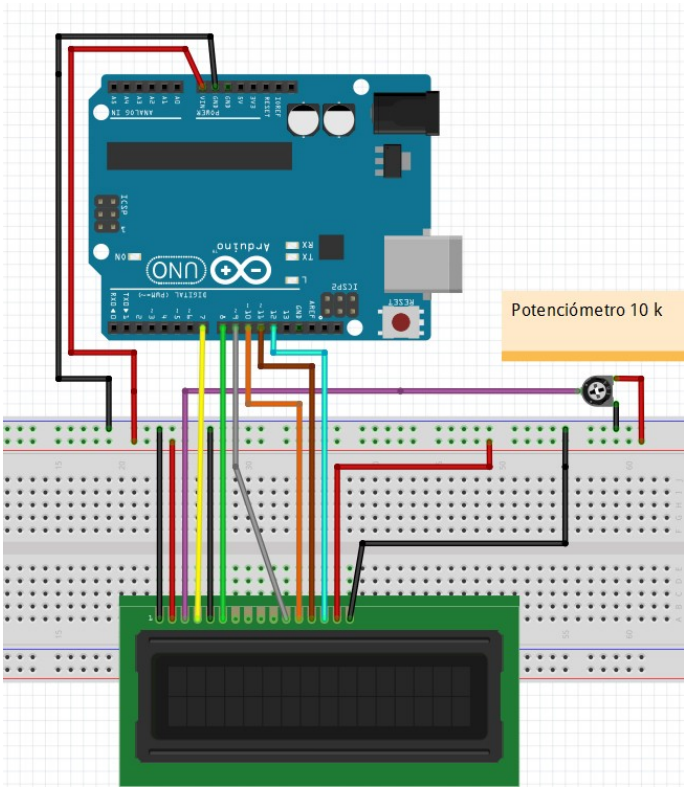
La pantalla de cristal líquido nos sirve para visualizar mensajes de texto o datos. Primero cargaremos la librería que la controla y le indicamos los pin que utilizará arduino.

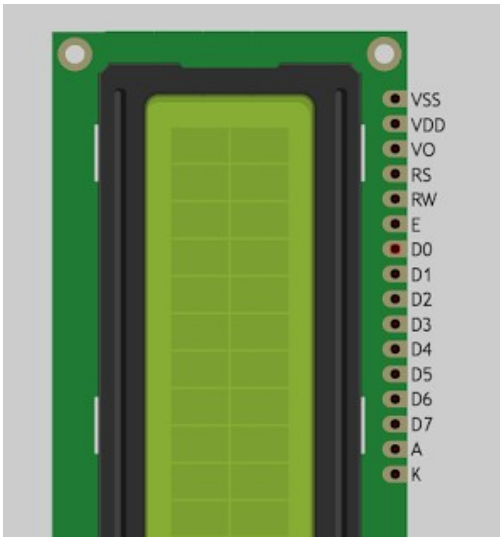
```
#include <LiquidCrystal.h>

LiquidCrystal lcd(7, 8, 9, 10, 11 , 12);

void setup() {
  lcd.begin(16, 2);
  lcd.setCursor(4,1);

  lcd.write("IES ZOCO ");
}
void loop() { }
```

// pines de arduino que se van a utilizar arduino

//tipo de LCD que usamos de 16 columnas y 2 filas
//indicamos el punto donde comenzamos a escribir columna 4 fila 1

// Texto que se escribe en la LCD



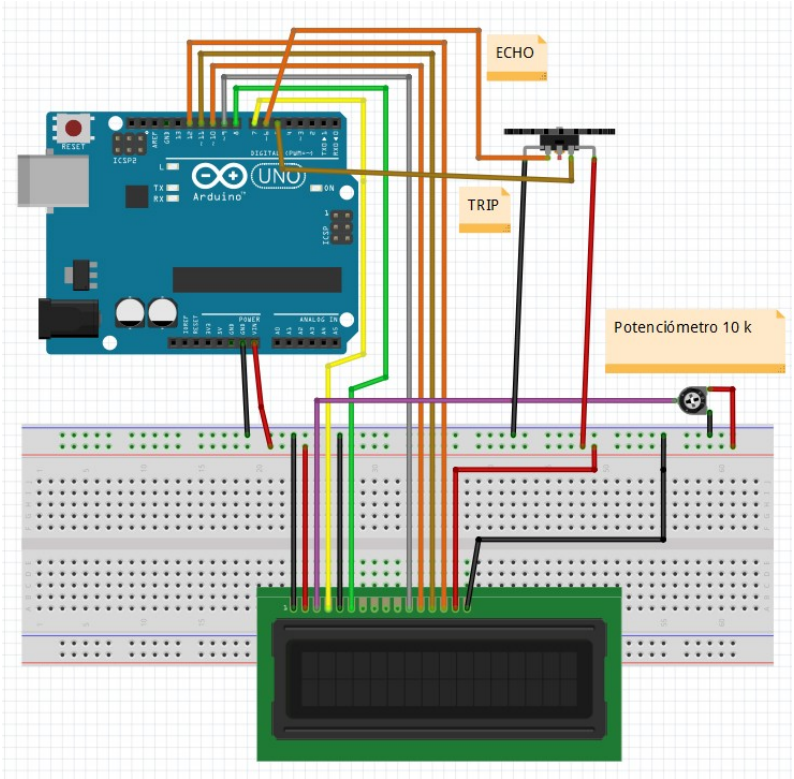


- **VSS** 0 volts o **GND.**
- **VDD** es la alimentación principal de la pantalla y el chip, lleva **5v**
- **VO** es el contraste de la pantalla, debe conectarse con un potenciometro de unos 10k ohms.
- **RS** es el selector de registro (el microcontrolador le comunica a la LCD si quiere mostrar caracteres o si lo que quiere es enviar comandos de control, como cambiar posición del cursor o borrar la pantalla, por ejemplo). En nuestro caso al **pin 6** de arduino.
- **RW** es el pin que comanda la lectura/escritura. En nuestro caso siempre estará en **GND** para que escriba en todo momento.
- **E** es enable, habilita la pantalla para recibir información al **pin 7** de arduino
- D0~D3 no los vamos a utilizar. Como pueden ver la pantalla tiene un bus de datos de 8 bits, de D0 a D7. Nosotros **solamente utilizaremos 4 bits, de D4 a D7**, que nos servirán para establecer las líneas de comunicación por donde se transfieren los datos.
- A y K son los pines del led de la luz de fondo de la pantalla. Se **conecta A a 5 volts y K a 0 v.**

PRÁCTICA 19. Uso de una pantalla LCD de cristal líquido

```
#include <Ultrasonic.h>
#include <LiquidCrystal.h>

LiquidCrystal lcd(7, 8, 9, 10, 11 , 12);
Ultrasonic ultrasonic(5,6,3000);


void setup() {
lcd.begin(16, 2);

lcd.print("medida longitud");

delay(3000); //espera antes de borrar
lcd.clear();

}
void loop()
{
  lcd.setCursor(6, 2);
  lcd.print(ultrasonic.Ranging(CM));
lcd.print("cm");
delay(200);
  lcd.clear();
}
```

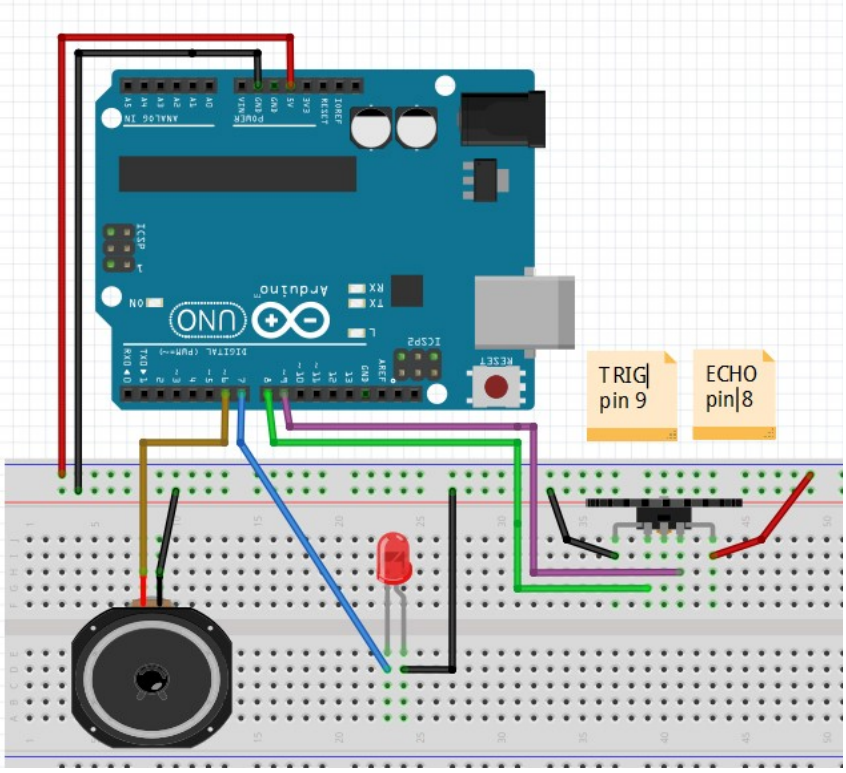
```
// incluimos la librería que controla el emisor de ultrasonidos.
// incluimos la librería que controla la pantalla LCD

// pin de arduino que se conectan la LCD
// (Trig al PIN 5,Echo al PIN  6 , milisegundos máximos)



// tipo de pantalla que tenemos de 16 columnas y 2 filas
//imprime este texto al comenzar el programa, luego lo borra y no
vuelve a escribirlo porque se va a ejecutar solo el void loop

                                        // se situa el cursor en la columna 6 fila 2 de la LCD

                                        // imprime los cm de la medida
                                        // imprime la palabra " cm" después de la cantidad

                                        //borra la pantalla
```



Añadimos al circuito el emisor de ultrasonidos:
GND a 0v; Vcc a 5v; Echo al pin 6 de arcuino y Trip al pin 5 de arduino.

PRÁCTICA 20. Sensor de aparcamiento de un coche con zumbador

Vamos a medir la distancia a un objeto y la visualizaremos. Si esta distancia es mayor de 70 cm el zumbador no sonará. Si la distancia está entre 50 y 70 cm sonará y se apagará el zumbador y se encenderá el led a una velocidad lenta. Conforme se acerque el obstáculo va aumentando la velocidad en que suena.



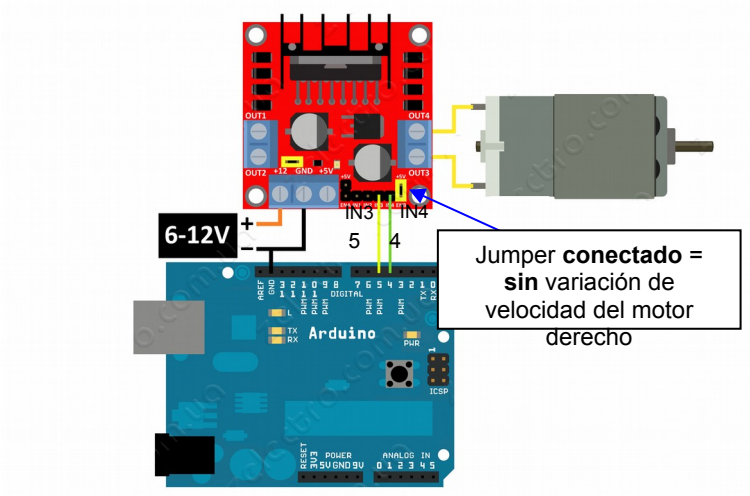

```
#include <Ultrasonic.h>

Ultrasonic ultrasonic(9,8,30000); // Trig 9 y Echo al pin 8, microsegundos máximos que mide el emisor 30000 ms= 5 metros. CUANTO MAS CERCA ESTÉ EL OBJETO MENOR SERÁ EL TIEMPO DE REGRESO DEL ULTRASONIDO
int led=7;
int zumbador=6;
long distancia =0;
int pausa=1000;

void setup() {
  Serial.begin(9600);                    //comenzamos la comunicación serial
  pinMode(zumbador, OUTPUT);
  pinMode(led, OUTPUT);
}
 void loop() {

  distancia=ultrasonic.Ranging(CM);

  if(distancia>50 && distancia<70) {          //si la distancia es mayor de 50 cm y menor de 70 pita lentamente

    pausa = distancia * 15;
    digitalWrite(zumbador, HIGH);
    digitalWrite(led,HIGH);
    delay(pausa);
  }
  else if(distancia<50 && distancia>20){ //si la distancia está comprendida entre 20 cm y 50 cm pita más rápido
```

```
  pausa = distancia * 10;
  digitalWrite(zumbador, HIGH);
  digitalWrite(led,HIGH);
  delay(pausa);
  }
  else if(distancia<20){    //si la distancia es menor de 20 cm aumenta la velocidad de parpadeo pita muy rápido
  pausa = distancia * 5;
  digitalWrite (zumbador, HIGH);
  digitalWrite(led,HIGH);
  delay(pausa);
  }
  analogWrite(zumbador,LOW);
  digitalWrite(led,LOW);
  delay(pausa);

  Serial.print("distancia =     "); // imprime un texto entrecomillado (tiempo= )
  Serial.print(ultrasonic.Ranging(CM));    //imprime el valor de la variable distancia al objeto
  Serial.println("        cm" );          // impirme un texto entrecomillado (cm) print LN significa que cambie de linea cada lectura

  delay(1000);                          // Espera 1 segundo.
}
```

PRÁCTICA 21. Control de un motor DC con el regulador **L298N**

Como demostración, vamos a controlar un motor DC a través de la salida B del módulo. El pin **ENB** se conectará con el jumper a +5V, para indicar que no vamos a controlar la velocidad del motor.

Esquema de conexión



El programa básicamente activa el motor en un sentido por 4 segundos, luego detiene el motor por 0.5 segundos, después activa el motor en sentido inverso por 4 segundos y por último detiene el motor por 5 segundos. Luego repite la acción indefinidamente.

Alimentamos Arduino por Vin (+5 v) y GND (0V)

Conectamos el jumper en esta posición indicando que sólo vamos a controlar la dirección de giro del motor y **NO su velocidad.**

Conectamos salida 5 de arduino con entrada IN3 y salida 4 con IN4.

Programa Arduino:

1º Indicamos las variables que son un valor que arduino guarda en su memoria. Creamos la variable entera IN3 que vale 5 y la variable IN4 que vale 4.

2º Configuramos arduino y le indicamos que pines son entradas y cuales salidas. En este caso IN4 (o sea 5) va a ser una salida. Todo esto se hace en el "void setup()" dentro de los corchetes { }.

3º Por último escribimos las instrucciones que deberá repetir Arduino de forma repetitiva. En este caso:

a) Escribe o pone (*digitalWrite*) a 5 voltios la variable IN4 y a 0v la variable IN5, y espera (*delay*) 4 segundos. Con esto logramos tener 5v en una entrada del motor y 0 voltios en la otra y el motor girará en una dirección.
b) Colocamos ambas entradas a 0 v y el motor se para 5 segundos
c) Ahora cambiamos y ponemos a 0v la primera y a 5 v la segunda con lo que el motor gira en la otra dirección.
d) Colocamos ambas entradas a 0 v y el motor se para 5 segundos.

Código en Arduino

```
/*
 Ejemplo de control de motor DC usando modulo L298
 El programa activa el motor en un sentido por 4 segundos,
 para el motor por 500 ms, activa el motor en sentido inverso por 4 segundos
 y se detiene por 5 segundos. Luego repite la acción indefinidamente. */
int IN3 = 5;
int IN4 = 4;
void setup()
{
pinMode (IN3, OUTPUT); // Conectamos la salida 5 de arduino con la entrada IN3 de la placa
pinMode (IN4, OUTPUT); // Conectamos la salida 4 de arduino con la entrada IN4 de la placa.
}
void loop()
```

```
{
  digitalWrite (IN4, HIGH); // Motor gira en un sentido, al poner una entrada a 5 v y otra a 0 v
  digitalWrite (IN3, LOW);
  delay(4000);

  digitalWrite (IN4, LOW);  // Motor no gira, se para. Ambas entradas a 0v
  delay(5000);


  digitalWrite (IN3, HIGH); //Motor gira en sentido inverso, al cambiar la entrada que está a 5v
  delay(4000);

  digitalWrite (IN3, LOW); // Motor no gira, se para. Ambas entradas a 0v
  delay(5000);
}
```

PRÁCTICA 21. Activación de una salida ANALÓGICA de 0 y 5 v PWM

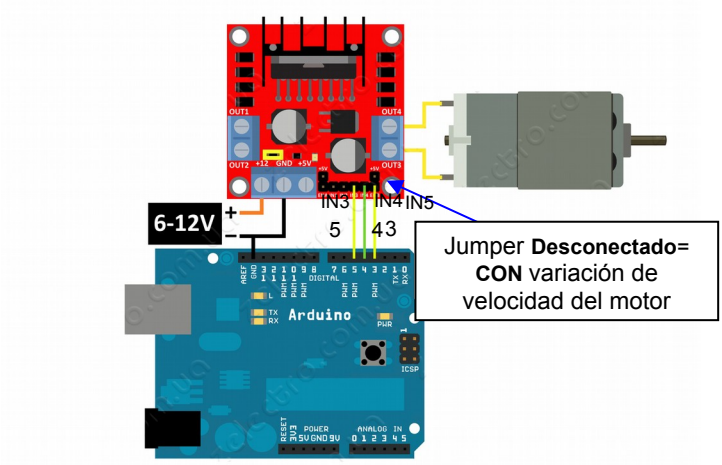
Podemos hacer que Arduino coloque en un pin determinado un valor que no sea 0 ó 5 voltios (0 ó 1), sino que sea un valor de tensión comprendido entre 0 y 5 v, por ejemplo 1v, 2v, 3v,...Vamos a utilizar esto para cambiar la velocidad de un motor o la luz de un led. Si utilizamos un motor NO debemos frenar el eje para no quemar Arduino.

Las salidas analógicas funcionan mucho mejor con reguladores de velocidad que veremos en las últimas prácticas.

int Motor=12;
int t=2000;
void setup()
{
pinMode(Motor, OUTPUT);
}
void loop()
{
analogWrite(Motor, 130);
delay(t);
analogWrite(Motor, 0);
delay(t);
analogWrite(Motor, 255);
delay(t);
analogWrite(Motor, 0);
delay(t);
}
	Ahora le decimos a Arduino que la variable “Motor” es 12 y la variable “t” es 2000.
	setup,
	indicamos que “Motor” es una salida (o sea que el pin 12 es una salida).
	loop
	analogWrite
	Escribe en “Motor” un valor analógico de 2´5 voltios y espera un tiempo t. y el motor irá más lento.
	Luego para el motor y espera.
	Escribe en “Motor” un valor analógico de 5 voltios y espera un tiempo t. El motor irá a máxima velocidad.
	Luego para el motor y espera.Repite el programa indefinidamente.

PRÁCTICA 23. Control de un motor DC variando su velocidad con el regulador **L298N**

Si queremos controlar la velocidad del motor, tenemos que hacer uso de PWM. Este PWM será aplicado a los pines de activación de cada salida o pines ENA y ENB respectivamente, por tanto los jumper de selección no serán usados.



El programa básicamente activa el motor en un sentido por 4 segundos, luego detiene el motor por 0.5 segundos, después activa el motor en sentido inverso por 4 segundos y por último detiene el motor por 5 segundos. Luego repite la acción indefinidamente.

Código en Arduino

El programa controla la velocidad de un motor DC aplicando PWM al pin ENB del módulo L298N.

```
/*
 Ejemplo de control de motor DC usando modulo L298. Utilizamos una salida analógica variable entre 0 y 5 voltios, de esta forma podremos variar la velocidad del motor que irá cada vez más rápido.
 */
int IN3 = 5;    // Input3 conectada al pin 5
int IN4 = 4;    // Input4 conectada al pin 4
int ENB = 3;    // ENB conectada al pin 3 de Arduino
void setup()
{
 pinMode (ENB, OUTPUT);
 pinMode (IN3, OUTPUT);
 pinMode (IN4, OUTPUT);
}
void loop()
{
  digitalWrite (IN3, HIGH); //Preparamos la salida para que el motor gire en un sentido
  digitalWrite (IN4, LOW);
  analogWrite(ENB,55); // Escribimos en el pin ENB un valor de tensión de 1 v. El motor gira despacio.
  delay(2000);  //Espera 2 segundos
  analogWrite(ENB,105); // Escribimos en el pin ENB un valor de tensión de 2 v. El motor gira  más rápido
  delay(2000); //espera
  analogWrite(ENB,255); //Escribimos en el pin ENB un valor de 5 v. El motor va a máxima velocidad.
  delay(2000); //Espera 2 segundos
  analogWrite(ENB,0); // Apagamos el motor
  delay(5000); //Espera 5 segundos
}
```

PRÁCTICA 23. Programa básico de control del robot controlado por un sensor de ultrasonidos.

El robot envía por el pin TRIG del sensor de ultrasonidos un pulso de 10 milisegundos y almacena en la variable *duración* el tiempo que tarda el pulso o señal de ultrasonidos en regresar. Para transformar en centímetros este tiempo lo dividimos entre dos (porque es un tiempo de ida y de vuelta al robot) y luego entre 29. Cuando ya conocemos la distancia en centímetros, almacenamos esta información en la variable *distancia.*

Con la orden IF determinamos que, si la distancia es mayor de 2 cm y menor de 15 centímetros el robot debe parar motores, retroceder y girar. Si no detecta ningún objeto a menos de 15 cm, avanzará de nuevo.

```
int ECHO = 8;            // define el pin 2 como (echo) para el Ultrasonido
int TRIG = 9;            // define el pin 3 como (trig) para el Ultrasonido
int duracion             //  Creamos al variable donde guardaremos el tiempo que tarda en regresa el ultrasonido
int distancia;           //  Creamos al variable donde guardaremos la distancia que hemos medido

int MderA = 4;    // Llamamos MderA a conector IN3 del motor derecho que conectamos al pin 4 de Arduino.
int MderB = 3;    // Llamamos MderB a conector IN4 del motor izquierdo que conectamos al pin 3 de Arduino
int MderVelo = 2;    // Llamamos MderVel a conector ENB que controla la velocidad del motor derecho y conectamos al pin2

int MizqB = 5;    // Llamamos MizqB a conector IN2 del motor izquierdo que conectamos al pin 5 de Arduino.
int MizqA = 6;    // Llamamos MizqA a conector IN1 del motor izquierdo que conectamos al pin 6 de Arduino
int MizqVelo = 7;    // Llamamos MizqVel a conector ENA que controla la velocidad del motor izquierdo y conectamos al pin7

void setup() {
  pinMode(ECHO, INPUT);          // define el pin 8 como entrada de Arduino procedente del ultrasonido
  pinMode(TRIG,OUTPUT);       // define el pin 3 como salida de Arduino que va al ultrasonido

 pinMode (MderA, OUTPUT);     // Define como salidas de Arduino todos esos pines de comunicación
 pinMode (MderB, OUTPUT);       //Que van a controlar los motores
 pinMode (MderVelo, OUTPUT);
 pinMode (MizqB, OUTPUT);
 pinMode (MizqA, OUTPUT);
 pinMode (MizqVelo, OUTPUT);
}

void loop() {
  digitalWrite(TRIG, HIGH);   // Lanza un pulso por el Trigger del ultrasonidos durante 10 milisegundos
     delay(10);
     digitalWrite(TRIG, LOW);                // anula el pulso  al pasar los 10 milisegundos
     duracion = pulseIn(ECHO, HIGH);  // Lee el tiempo del Echo, que es el que ha tardado en regresar el pulso que
hemos lanzado y lo guarda en la variable "duración".

     distancia = (duracion/2) / 29;       // calcula la distancia en centímetros y la guarda en variable distancia. Para ello
la divide entre 2 (porque es de ida y vuelta) y entre 29 (para pasarlo a centímetros) guarda el resultado en la variable
"distancia"

 delay(10);

 if (distancia <= 15 && distancia >=2){   //si la distancia es menor de 15 cm y mayor de 2 cm entonces

  digitalWrite (MderB, LOW);            //PARAMOS motores
  digitalWrite (MderA, LOW);
  digitalWrite (MizqB, LOW);
  digitalWrite (MizqA, LOW);
  delay(200);
   digitalWrite (MderB, LOW);      //RETROCEDE
   digitalWrite (MderA, HIGH);
  analogWrite(MderVelo, 100); // hacemos que uno gire a mayor velocidad que otro para que gires  rápido.
  digitalWrite (MizqB, LOW); //marcha atrás
```

```
  digitalWrite (MizqA, HIGH);
  analogWrite(MderVelo, 100); // hacemos que uno gire a mayor velocidad que otro para que gires  rápido.
  delay(700);
     digitalWrite (MderB, LOW);    //GIRO
  digitalWrite (MderA, HIGH);
  analogWrite(MderVelo, 200); // hacemos que uno gire a mayor velocidad que otro para que gires  rápido.
  digitalWrite (MizqB, LOW); //marcha atrás
  digitalWrite (MizqA, LOW);
   delay(150);
 }
   else{

   digitalWrite (MderB, HIGH); //Ambos motores vuelven a girar hacia delante a máxima potencia
  digitalWrite (MderA, LOW);
  analogWrite(MderVelo,160);
  digitalWrite (MizqB, HIGH);
  digitalWrite (MizqA, LOW);
  analogWrite(MizqVelo,160);

    }
}
```

PRÁCTICA 11. Motor que gira a derechas e izquierdas variando su velocidad

Creamos 4 variables una para que el motor gire a derechas, otra a izquierdas y otras para que gire lentamente. Después podemos hacer combinaciones

```
int rojo = 12;          // Crea una variable llamada "rojo" y le asigna un valor 12
int negro = 10;
void setup()
{
  pinMode(rojo, OUTPUT); // Indica que la variable "rojo" es una salida de arduino, y por tanto el pin 12 es una salida.
  pinMode(negro, OUTPUT);
}
void loop()
{
  Motor_derecha();          // ejecuta la función MOTOR GIRA A LA DERECHA

  M_izquierda_lento();      // ejecuta la función MOTOR IZQUIERDAS LENTAMENTE

   Motor_izquierda();          // ejecuta la función MOTOR GIRA A LA IZQUIERDA

M_derecha_lento();          // ejecuta la función MOTOR DERECHA LENTAMENTE
 }
void Motor_derecha()    // Crea la función motor gira a derechas
{
 digitalWrite(rojo, HIGH);       // EL MOTOR GIRA A LA DERECHA. Escribe en la variable "rojo" un valor alto, un "1".
  digitalWrite(negro, LOW);      // Escribe en la variable "negro" un valor alto, un "0".
  delay(2000);                   // Espera 2 segundos
  digitalWrite(rojo, LOW);       // Escribe en la variable "rojo" un valor bajo, un "0".
 delay(500);
}
void Motor_izquierda()     // Crea la función motor gira a izquierdas
{
 digitalWrite(negro, HIGH);     // EL MOTOR GIRA A LA IZQUIERDA.
  digitalWrite(rojo, LOW);
  delay(2000);
```

```
  digitalWrite(negro, LOW);
 delay(500);
}
void M_derecha_lento()   // Crea la función motor gira a derechas lento
{
analogWrite(rojo, 130);          // el motor gira a derechas lentamente.
  digitalWrite(negro, LOW);
  delay(2000);
  digitalWrite(rojo, LOW);
 delay(500);
}
void M_izquierda_lento()   // Crea la función motor gira a izquierdas lento
{
analogWrite(negro, 130);   // el motor gira a izquierdas lentamente.
  digitalWrite(rojo, LOW);
  delay(2000);
  digitalWrite(negro, LOW);
 delay(500);
}
```